

FULL MULTIPROCESSOR SPECULATION MECHANISM IN A SYMMETRIC  
MULTIPROCESSOR (SMP) SYSTEM

BACKGROUND OF THE INVENTION

1. Technical Field:

5 The present invention relates in general to data processing systems and in particular to instruction processing in multiprocessor data processing systems. Still more particularly, the present invention relates to a method and processor architecture for improving processing efficiency by enabling full, un-throttled execution of instructions beyond barrier operations.

10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120 125 130 135 140 145 150 155 160 165 170 175 180 185 190 195 200 205 210 215 220 225 230 235 240 245 250 255 260 265 270 275 280 285 290 295 300 305 310 315 320 325 330 335 340 345 350 355 360 365 370 375 380 385 390 395 400 405 410 415 420 425 430 435 440 445 450 455 460 465 470 475 480 485 490 495 500 505 510 515 520 525 530 535 540 545 550 555 560 565 570 575 580 585 590 595 600 605 610 615 620 625 630 635 640 645 650 655 660 665 670 675 680 685 690 695 700 705 710 715 720 725 730 735 740 745 750 755 760 765 770 775 780 785 790 795 800 805 810 815 820 825 830 835 840 845 850 855 860 865 870 875 880 885 890 895 900 905 910 915 920 925 930 935 940 945 950 955 960 965 970 975 980 985 990 995 1000 1005 1010 1015 1020 1025 1030 1035 1040 1045 1050 1055 1060 1065 1070 1075 1080 1085 1090 1095 1100 1105 1110 1115 1120 1125 1130 1135 1140 1145 1150 1155 1160 1165 1170 1175 1180 1185 1190 1195 1200 1205 1210 1215 1220 1225 1230 1235 1240 1245 1250 1255 1260 1265 1270 1275 1280 1285 1290 1295 1300 1305 1310 1315 1320 1325 1330 1335 1340 1345 1350 1355 1360 1365 1370 1375 1380 1385 1390 1395 1400 1405 1410 1415 1420 1425 1430 1435 1440 1445 1450 1455 1460 1465 1470 1475 1480 1485 1490 1495 1500 1505 1510 1515 1520 1525 1530 1535 1540 1545 1550 1555 1560 1565 1570 1575 1580 1585 1590 1595 1600 1605 1610 1615 1620 1625 1630 1635 1640 1645 1650 1655 1660 1665 1670 1675 1680 1685 1690 1695 1700 1705 1710 1715 1720 1725 1730 1735 1740 1745 1750 1755 1760 1765 1770 1775 1780 1785 1790 1795 1800 1805 1810 1815 1820 1825 1830 1835 1840 1845 1850 1855 1860 1865 1870 1875 1880 1885 1890 1895 1900 1905 1910 1915 1920 1925 1930 1935 1940 1945 1950 1955 1960 1965 1970 1975 1980 1985 1990 1995 2000 2005 2010 2015 2020 2025 2030 2035 2040 2045 2050 2055 2060 2065 2070 2075 2080 2085 2090 2095 2100 2105 2110 2115 2120 2125 2130 2135 2140 2145 2150 2155 2160 2165 2170 2175 2180 2185 2190 2195 2200 2205 2210 2215 2220 2225 2230 2235 2240 2245 2250 2255 2260 2265 2270 2275 2280 2285 2290 2295 2300 2305 2310 2315 2320 2325 2330 2335 2340 2345 2350 2355 2360 2365 2370 2375 2380 2385 2390 2395 2400 2405 2410 2415 2420 2425 2430 2435 2440 2445 2450 2455 2460 2465 2470 2475 2480 2485 2490 2495 2500 2505 2510 2515 2520 2525 2530 2535 2540 2545 2550 2555 2560 2565 2570 2575 2580 2585 2590 2595 2600 2605 2610 2615 2620 2625 2630 2635 2640 2645 2650 2655 2660 2665 2670 2675 2680 2685 2690 2695 2700 2705 2710 2715 2720 2725 2730 2735 2740 2745 2750 2755 2760 2765 2770 2775 2780 2785 2790 2795 2800 2805 2810 2815 2820 2825 2830 2835 2840 2845 2850 2855 2860 2865 2870 2875 2880 2885 2890 2895 2900 2905 2910 2915 2920 2925 2930 2935 2940 2945 2950 2955 2960 2965 2970 2975 2980 2985 2990 2995 3000 3005 3010 3015 3020 3025 3030 3035 3040 3045 3050 3055 3060 3065 3070 3075 3080 3085 3090 3095 3100 3105 3110 3115 3120 3125 3130 3135 3140 3145 3150 3155 3160 3165 3170 3175 3180 3185 3190 3195 3200 3205 3210 3215 3220 3225 3230 3235 3240 3245 3250 3255 3260 3265 3270 3275 3280 3285 3290 3295 3300 3305 3310 3315 3320 3325 3330 3335 3340 3345 3350 3355 3360 3365 3370 3375 3380 3385 3390 3395 3400 3405 3410 3415 3420 3425 3430 3435 3440 3445 3450 3455 3460 3465 3470 3475 3480 3485 3490 3495 3500 3505 3510 3515 3520 3525 3530 3535 3540 3545 3550 3555 3560 3565 3570 3575 3580 3585 3590 3595 3600 3605 3610 3615 3620 3625 3630 3635 3640 3645 3650 3655 3660 3665 3670 3675 3680 3685 3690 3695 3700 3705 3710 3715 3720 3725 3730 3735 3740 3745 3750 3755 3760 3765 3770 3775 3780 3785 3790 3795 3800 3805 3810 3815 3820 3825 3830 3835 3840 3845 3850 3855 3860 3865 3870 3875 3880 3885 3890 3895 3900 3905 3910 3915 3920 3925 3930 3935 3940 3945 3950 3955 3960 3965 3970 3975 3980 3985 3990 3995 4000 4005 4010 4015 4020 4025 4030 4035 4040 4045 4050 4055 4060 4065 4070 4075 4080 4085 4090 4095 4100 4105 4110 4115 4120 4125 4130 4135 4140 4145 4150 4155 4160 4165 4170 4175 4180 4185 4190 4195 4200 4205 4210 4215 4220 4225 4230 4235 4240 4245 4250 4255 4260 4265 4270 4275 4280 4285 4290 4295 4300 4305 4310 4315 4320 4325 4330 4335 4340 4345 4350 4355 4360 4365 4370 4375 4380 4385 4390 4395 4400 4405 4410 4415 4420 4425 4430 4435 4440 4445 4450 4455 4460 4465 4470 4475 4480 4485 4490 4495 4500 4505 4510 4515 4520 4525 4530 4535 4540 4545 4550 4555 4560 4565 4570 4575 4580 4585 4590 4595 4600 4605 4610 4615 4620 4625 4630 4635 4640 4645 4650 4655 4660 4665 4670 4675 4680 4685 4690 4695 4700 4705 4710 4715 4720 4725 4730 4735 4740 4745 4750 4755 4760 4765 4770 4775 4780 4785 4790 4795 4800 4805 4810 4815 4820 4825 4830 4835 4840 4845 4850 4855 4860 4865 4870 4875 4880 4885 4890 4895 4900 4905 4910 4915 4920 4925 4930 4935 4940 4945 4950 4955 4960 4965 4970 4975 4980 4985 4990 4995 5000 5005 5010 5015 5020 5025 5030 5035 5040 5045 5050 5055 5060 5065 5070 5075 5080 5085 5090 5095 5100 5105 5110 5115 5120 5125 5130 5135 5140 5145 5150 5155 5160 5165 5170 5175 5180 5185 5190 5195 5200 5205 5210 5215 5220 5225 5230 5235 5240 5245 5250 5255 5260 5265 5270 5275 5280 5285 5290 5295 5300 5305 5310 5315 5320 5325 5330 5335 5340 5345 5350 5355 5360 5365 5370 5375 5380 5385 5390 5395 5400 5405 5410 5415 5420 5425 5430 5435 5440 5445 5450 5455 5460 5465 5470 5475 5480 5485 5490 5495 5500 5505 5510 5515 5520 5525 5530 5535 5540 5545 5550 5555 5560 5565 5570 5575 5580 5585 5590 5595 5600 5605 5610 5615 5620 5625 5630 5635 5640 5645 5650 5655 5660 5665 5670 5675 5680 5685 5690 5695 5700 5705 5710 5715 5720 5725 5730 5735 5740 5745 5750 5755 5760 5765 5770 5775 5780 5785 5790 5795 5800 5805 5810 5815 5820 5825 5830 5835 5840 5845 5850 5855 5860 5865 5870 5875 5880 5885 5890 5895 5900 5905 5910 5915 5920 5925 5930 5935 5940 5945 5950 5955 5960 5965 5970 5975 5980 5985 5990 5995 6000 6005 6010 6015 6020 6025 6030 6035 6040 6045 6050 6055 6060 6065 6070 6075 6080 6085 6090 6095 6100 6105 6110 6115 6120 6125 6130 6135 6140 6145 6150 6155 6160 6165 6170 6175 6180 6185 6190 6195 6200 6205 6210 6215 6220 6225 6230 6235 6240 6245 6250 6255 6260 6265 6270 6275 6280 6285 6290 6295 6300 6305 6310 6315 6320 6325 6330 6335 6340 6345 6350 6355 6360 6365 6370 6375 6380 6385 6390 6395 6400 6405 6410 6415 6420 6425 6430 6435 6440 6445 6450 6455 6460 6465 6470 6475 6480 6485 6490 6495 6500 6505 6510 6515 6520 6525 6530 6535 6540 6545 6550 6555 6560 6565 6570 6575 6580 6585 6590 6595 6600 6605 6610 6615 6620 6625 6630 6635 6640 6645 6650 6655 6660 6665 6670 6675 6680 6685 6690 6695 6700 6705 6710 6715 6720 6725 6730 6735 6740 6745 6750 6755 6760 6765 6770 6775 6780 6785 6790 6795 6800 6805 6810 6815 6820 6825 6830 6835 6840 6845 6850 6855 6860 6865 6870 6875 6880 6885 6890 6895 6900 6905 6910 6915 6920 6925 6930 6935 6940 6945 6950 6955 6960 6965 6970 6975 6980 6985 6990 6995 7000 7005 7010 7015 7020 7025 7030 7035 7040 7045 7050 7055 7060 7065 7070 7075 7080 7085 7090 7095 7100 7105 7110 7115 7120 7125 7130 7135 7140 7145 7150 7155 7160 7165 7170 7175 7180 7185 7190 7195 7200 7205 7210 7215 7220 7225 7230 7235 7240 7245 7250 7255 7260 7265 7270 7275 7280 7285 7290 7295 7300 7305 7310 7315 7320 7325 7330 7335 7340 7345 7350 7355 7360 7365 7370 7375 7380 7385 7390 7395 7400 7405 7410 7415 7420 7425 7430 7435 7440 7445 7450 7455 7460 7465 7470 7475 7480 7485 7490 7495 7500 7505 7510 7515 7520 7525 7530 7535 7540 7545 7550 7555 7560 7565 7570 7575 7580 7585 7590 7595 7600 7605 7610 7615 7620 7625 7630 7635 7640 7645 7650 7655 7660 7665 7670 7675 7680 7685 7690 7695 7700 7705 7710 7715 7720 7725 7730 7735 7740 7745 7750 7755 7760 7765 7770 7775 7780 7785 7790 7795 7800 7805 7810 7815 7820 7825 7830 7835 7840 7845 7850 7855 7860 7865 7870 7875 7880 7885 7890 7895 7900 7905 7910 7915 7920 7925 7930 7935 7940 7945 7950 7955 7960 7965 7970 7975 7980 7985 7990 7995 8000 8005 8010 8015 8020 8025 8030 8035 8040 8045 8050 8055 8060 8065 8070 8075 8080 8085 8090 8095 8100 8105 8110 8115 8120 8125 8130 8135 8140 8145 8150 8155 8160 8165 8170 8175 8180 8185 8190 8195 8200 8205 8210 8215 8220 8225 8230 8235 8240 8245 8250 8255 8260 8265 8270 8275 8280 8285 8290 8295 8300 8305 8310 8315 8320 8325 8330 8335 8340 8345 8350 8355 8360 8365 8370 8375 8380 8385 8390 8395 8400 8405 8410 8415 8420 8425 8430 8435 8440 8445 8450 8455 8460 8465 8470 8475 8480 8485 8490 8495 8500 8505 8510 8515 8520 8525 8530 8535 8540 8545 8550 8555 8560 8565 8570 8575 8580 8585 8590 8595 8600 8605 8610 8615 8620 8625 8630 8635 8640 8645 8650 8655 8660 8665 8670 8675 8680 8685 8690 8695 8700 8705 8710 8715 8720 8725 8730 8735 8740 8745 8750 8755 8760 8765 8770 8775 8780 8785 8790 8795 8800 8805 8810 8815 8820 8825 8830 8835 8840 8845 8850 8855 8860 8865 8870 8875 8880 8885 8890 8895 8900 8905 8910 8915 8920 8925 8930 8935 8940 8945 8950 8955 8960 8965 8970 8975 8980 8985 8990 8995 9000 9005 9010 9015 9020 9025 9030 9035 9040 9045 9050 9055 9060 9065 9070 9075 9080 9085 9090 9095 9100 9105 9110 9115 9120 9125 9130 9135 9140 9145 9150 9155 9160 9165 9170 9175 9180 9185 9190 9195 9200 9205 9210 9215 9220 9225 9230 9235 9240 9245 9250 9255 9260 9265 9270 9275 9280 9285 9290 9295 9300 9305 9310 9315 9320 9325 9330 9335 9340 9345 9350 9355 9360 9365 9370 9375 9380 9385 9390 9395 9400 9405 9410 9415 9420 9425 9430 9435 9440 9445 9450 9455 9460 9465 9470 9475 9480 9485 9490 9495 9500 9505 9510 9515 9520 9525 9530 9535 9540 9545 9550 9555 9560 9565 9570 9575 9580 9585 9590 9595 9600 9605 9610 9615 9620 9625 9630 9635 9640 9645 9650 9655 9660 9665 9670 9675 9680 9685 9690 9695 9700 9705 9710 9715 9720 9725 9730 9735 9740 9745 9750 9755 9760 9765 9770 9775 9780 9785 9790 9795 9800 9805 9810 9815 9820 9825 9830 9835 9840 9845 9850 9855 9860 9865 9870 9875 9880 9885 9890 9895 9900 9905 9910 9915 9920 9925 9930 9935 9940 9945 9950 9955 9960 9965 9970 9975 9980 9985 9990 9995 9999 10000 10005 10010 10015 10020 10025 10030 10035 10040 10045 10050 10055 10060 10065 10070 10075 10080 10085 10090 10095 10099 10100 10101 10102 10103 10104 10105 10106 10107 10108 10109 10110 10111 10112 10113 10114 10115 10116 10117 10118 10119 10120 10121 10122 10123 10124 10125 10126 10127 10128 10129 10130 10131 10132 10133 10134 10135 10136 10137 10138 10139 10140 10141 10142 10143 10144 10145 10146 10147 10148 10149 10150 10151 10152 10153 10154 10155 10156 10157 10158 10159 10160 10161 10162 10163 10164 10165 10166 10167 10168 10169 10170 10171 10172 10173 10174 10175 10176 10177 10178 10179 10180 10181 10182 10183 10184 10185 10186 10187 10188 10189 10190 10191 10192 10193 10194 10195 10196 10197 10198 10199 10200 10201 10202 10203 10204 10205 10206 10207 10208 10209 10210 10211 10212 10213 10214 10215 10216 10217 10218 10219 10220 10221 10222 10223 10224 10225 10226 10227 10228 10229 10230 10231 10232 10233 10234 10235 10236 10237 10238 10239 10240 10241 10242 10243 10244 10245 10246 10247 10248 10249 10250 10251 10252 10253 10254 10255 10256 10257 10258 10259 10260 10261 10262 10263 10264 10265 10266 10267 10268 10269 10270 10271 10272 10273 10274 10275 10276 10277 10278 10279 10280 10281 10282 10283 10284 10285 10286 10287 10288 10289 10290 10291 10292 10293 10294 10295 10296

order (i.e., later instructions executed before earlier instructions). However, due to possible data dependencies between groups, instructions in each group are executed in program order with respect to instructions in a next group to ensure correct processing results.

State-of-the-art superscalar processors provide a branch prediction mechanism by which branch instructions are permitted to be speculatively executed and later resolved. A superscalar processor may comprise, for example, an instruction cache for storing instructions, one or more execution units for executing sequential instructions, branch prediction and branch resolution logic for processing branch instructions, instruction sequencing logic for routing instructions to the various execution units, and registers for storing operands and result data.

When initially executed, conditional branch instructions are classified as unresolved. In order to minimize execution stalls, some processors speculatively execute unresolved branch instructions by predicting whether or not the indicated branch will be taken. Utilizing the result of the prediction, the instruction sequencing logic is then able to speculatively fetch instructions within a target execution path prior to the resolution of the branch. Presently, the more accurate branch prediction methodologies, such as branch history tables, yield correct predictions more than 92% of the time, which in terms of overall processor efficiency is widely considered to provide a significant improvement.

5

Typically, however, when a processor begins executing instructions within a speculatively predicted path (i.e., target or in-line path), processing of instructions within that path can only be completed up to the first barrier operation in the instruction sequence, and the processor waits until an acknowledgment is received for the barrier operation before continuing to process the instruction sequence down the branch path.

10

15

20

25

30

In multiprocessor systems, the correct completion of operations within code or instructions executing on a first processor may be dependent on operations on a second interconnected processor. For example, with load and store instructions executed by a load/store unit (LSU) of a first processor, a previous instruction that stores a value to a particular location must be executed before a later instruction that loads the value of that location.

Barrier instructions are placed within the instruction sequence to separate groups of instructions and ensure that all instructions within a first group are fully executed (i.e., the corresponding operations and results are visible to all other processors) before any instruction within a subsequent group is executed. The instruction set architecture (ISA) supported by most commercially available processors includes a barrier instruction, which initiates a barrier operation on the system. In the PowerPC™ family of processors, for example, one barrier instruction that is employed to establish a processing boundary is the "sync" instruction, and the corresponding transaction on the

5 system bus is called a synchronization operation (sync op). Other barrier instructions exist within the instruction set, but synch ops will be utilized generally within the present document to refer to global barrier instructions.

10

Barrier instructions are particularly necessary when the multiprocessor system includes superscalar processors supporting out-of-order instruction execution and weak memory consistency. However, there are implied barrier instructions utilized within in-order processor systems.

15

In slower processors, which operate at, for example, 100 MHz, each barrier instruction, such as a sync op, may require approximately 10 processor cycles to complete. In commercial server workloads, the sync ops typically degrade processing efficiency by approximately 5 percent. With faster processors, however, such as those operating in the Ghz range, a sync may complete in approximately 200 processor cycles and degrades processing efficiency by approximately 10 percent. Thus, syncs place a significant burden on processor efficiency, particularly because, in typical commercial software, syncs regularly occur every 500-1000 instructions. Each occurrence of a sync causes processors in a data processing system to be throttled for a lengthy time while the issuing processor waits on the sync operation to complete.

20

25

30

The inherent performance limitations of throttling the processor after each occurrence of a barrier instruction becomes even more acute with newer, high speed processor architectures, which have deep execution

5

pipelines, large instruction fetch latencies, and processes instructions with a high level of accuracy. Thus, throttling a processor from continuing along an execution path because of a barrier operation significantly limits processor efficiency.

10

The present invention recognizes that it would therefore be desirable to provide a method and processor architecture for enabling full processor speculation by executing all instructions beyond barrier operations to reduce processor throttling while waiting on a sync ack and thereby increase processor speed and efficiency.

## SUMMARY OF THE INVENTION

Described is a data processing system and processor that provides full multiprocessor speculation by which all instructions subsequent to barrier operations in a 5 instruction sequence are speculatively executed while the barrier operation is executing on the system bus (i.e., before the barrier operation completes and an acknowledgment is received at the issuing processor).

The processor comprises a the load/store unit (LSU) 10 having a barrier operation (BOP) controller that is coupled to and interacts with the LSU's load request queue (LRQ) and store/barrier queue. The BOP controller permits load instructions subsequent to syncs in an instruction sequence to be speculatively issued by the LRQ prior to the return of the sync acknowledgment. To 15 speculatively issue load requests, the barrier operation controller maintains a multiprocessor speculation (MS) flag in each entry of the LRQ. Load data returned by the speculative load request is immediately forwarded to the processor's execution units before the corresponding sync ack arrives and is utilized by the speculative processes 20 associated with the subsequent instructions.

Thus, instructions following an incompletely completed barrier 25 operation, such as a load/stores and other instructions, which may utilize the returned data are executed without throttling the processor by first determining if the barrier operation completes successfully. The processor thus continues processing instructions as if no speculation has occurred. The MS flag remains set in the

LRQ while the processor continues executing instructions. The MS flag is reset only when the sync ack is received. Because the speculative issuance of loads/stores beyond a barrier instruction have correct dependencies in over 99% of the times in high frequency processors, the processor continues to operate smoothly with an efficiency gain of up to 100 processor cycles when operating with full speculation.

In the preferred embodiment, the returned data and results of subsequent operations are held temporarily in the rename registers. A multiprocessor speculation flag is set in the corresponding rename registers to indicate that the value is speculative. When a barrier acknowledge is received by the BOP controller, the BOP controller messages logic affiliated with the processor's registers, which then resets the flag(s) of the corresponding rename register(s). The rename register is then characterized as the general purpose register (GPR) or floating point register (FPR) to which it is assigned.

In one preferred embodiment, the internal instruction set architecture (IISA) is provided one or more additional bits for utilization as the speculative flag(s). However, another preferred embodiment utilizes internal functionality of the associated queues to provide the additional bits that tag the particular instruction when the instruction is within the queue. In either case, as with the LRQ, the bit is reset when the sync ack returns.

In another embodiment, the invention permits embedded speculation by allowing speculative loads/stores and branch prediction to continue within the first and embedded instruction sequences.

5

The above as well as additional objects, features, and advantages of an illustrative embodiment will become apparent in the following detailed written description.

卷之三

**BRIEF DESCRIPTION OF THE DRAWINGS**

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself however, as well as a preferred mode of use, further objects and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

**Figure 1** depicts a high level block diagram of a processor and multiprocessor data processing system in which a preferred embodiment of the present invention may be advantageously implemented;

**Figure 2** is a block diagram of a preferred embodiment of a Load/Store Unit (LSU) utilized in accordance with the present invention;

**Figure 3** is a logic flow chart that illustrates the process of speculatively executing load instructions and subsequent instructions beyond syncs in accordance with the present invention;

**Figure 4** is a table representation of rename registers utilized in accordance with one embodiment of the present invention; and

**Figure 5** is a timing diagram illustrating processor cycles and instruction processing with full speculative execution in accordance with a preferred embodiment of the present invention.

## DETAILED DESCRIPTION OF AN ILLUSTRATIVE EMBODIMENT

With reference now to the figures, and in particular with reference to **Figure 1**, there is illustrated a high level block diagram of a processor within a multiprocessor data processing system in accordance with the present invention. As depicted, data processing system **8** includes two processors **10A** and **10B** (although additional processors are possible). Processors **10A**, **10B** preferably comprises one of the PowerPC™ line of microprocessors available from International Business Machines Corporation. Processors **10A**, **10B** are preferably superscalar with full out-of-order execution. Those skilled in the art will appreciate that other suitable processors can be utilized.

As illustrated, data processing system **8** further includes system memory **12**, which like processor **10B** is interconnected with processor **10A** via interconnect (or system bus) **14**. Interconnect **14**, which can comprise one or more buses or a cross-point switch, serves as a conduit for communication transactions between processors **10A**, **10B**, system memory **12**, and other devices associated with data processing system **8**. Each device coupled to interconnect **14** preferably snoops all communication transactions on interconnect **14**.

Processor **10A** is utilized for processing instructions and data in accordance with the present invention. Processor **10A** provides full-multiprocessor speculation by executing instructions (load/stores and

other instructions) following a barrier instruction in the instruction sequence before the corresponding barrier operation completes on the system bus and an acknowledgment is received.

5 Processor **10A** comprises a single integrated circuit superscalar processor, which, as discussed further below, includes various execution units, registers, buffers, memories, and other functional units that are all formed by integrated circuitry. Processor **10A** also includes an 10 on-chip multi-level cache hierarchy including a unified level two (L2) cache **16** and bifurcated level one (L1) instruction (I) and data (D) caches **18** and **20**, respectively. As is well-known to those skilled in the art, caches **16**, **18** and **20** provide low latency access to 15 cache lines corresponding to memory locations in system memory **12**.

20 Instructions are fetched for processing from L1 I-cache **18** in response to the effective address (EA) residing in instruction fetch address register (IFAR) **30**. During each cycle, a new instruction fetch address may be loaded into IFAR **30** from one of three sources: branch prediction unit (BPU) **36**, which provides speculative 25 target path addresses resulting from the prediction of conditional branch instructions, global completion table (GCT) **38**, which provides sequential path addresses, and branch execution unit (BEU) **92**, which in accordance with the present invention includes logic **160** that provides continuous speculative fetch and execution of

instructions along a predicted branch path beyond barrier operations.

If hit/miss logic **22** determines, after translation of the EA contained in IFAR **30** by effective-to-real address translation (ERAT) **32** and lookup of the real address (RA) in I-cache directory **34**, that the cache line of instructions corresponding to the EA in IFAR **30** does not reside in L1 I-cache **18**, then hit/miss logic **22** provides the RA to L2 cache **16** as a request address via I-cache request bus **24**. Such request addresses may also be generated by prefetch logic within L2 cache **16** based upon recent access patterns. In response to a request address, L2 cache **16** outputs a cache line of instructions, which are loaded into prefetch buffer (PB) **28** and L1 I-cache **18** via I-cache reload bus **26**, possibly after passing through optional predecode logic **144**.

Once the cache line specified by the EA in IFAR **30** resides in L1 cache **18**, L1 I-cache **18** outputs the cache line to both branch prediction unit (BPU) **36** and to instruction fetch buffer (IFB) **40**. BPU **36** scans the cache line of instructions for branch instructions and predicts an outcome of conditional branch instructions, if any. Such prediction can be performed by any known or future developed static or dynamic branch prediction methodology or, alternatively, can entail simply always selecting the next sequential address following the branch instruction (which is not truly branch prediction). In the preferred embodiment, branch prediction methodology utilizes branch history tables to

provide greater than 92% accuracy with branch prediction for commercial server workloads.

IFB **40** temporarily buffers the cache line of instructions received from L1 I-cache **18** until the cache line of instructions can be translated by instruction translation unit (ITU) **42**. In the illustrated embodiment of processor **10A**, ITU **42** translates instructions from user instruction set architecture (UISA) instructions into a possibly different number of internal ISA (IISA) instructions that are directly executable by the execution units of processor **10A**. Such translation may be performed, for example, by reference to microcode stored in a read-only memory (ROM) template. In at least some embodiments, the UISA-to-IISA translation results in a different number of IISA instructions than UISA instructions and/or IISA instructions of different lengths than corresponding UISA instructions. The resultant IISA instructions are then assigned by global completion table **38** to an instruction group, the members of which are permitted to be dispatched and executed out-of-order with respect to one another.

Each instruction group is separated by a barrier operation (or sync) as described in further detail below. Global completion table **38** tracks each instruction group for which execution has yet to be completed by at least one associated EA, which is preferably the EA of the oldest instruction in the instruction group.

Following UISA-to-IISA instruction translation, instructions are dispatched to one of latches **44**, **46**, **48**

and **50**, possibly out-of-order, based upon instruction type. That is, branch instructions and other condition register (CR) modifying instructions are dispatched to latch **44**, fixed-point and load-store instructions are dispatched to either of latches **46** and **48**, and floating-point instructions are dispatched to latch **50**. Each instruction requiring a rename register for temporarily storing execution results is then assigned one or more rename registers by the appropriate one of CR mapper **52**, link and count (LC) register mapper **54**, exception register (XER) mapper **56**, general-purpose register (GPR) mapper **58**, and floating-point register (FPR) mapper **60**. Utilization of the rename registers within the preferred embodiment of the invention is described below with reference to **Figure 4**.

Returning now to **Figure 1**, and particularly to processor **10A**, the dispatched instructions are then temporarily placed in an appropriate one of CR issue queue (CRIQ) **62**, branch issue queue (BIQ) **64**, fixed-point issue queues (FXIQs) **66** and **68**, and floating-point issue queues (FPIQs) **70** and **72**.

From issue queues **62**, **64**, **66**, **68**, **70** and **72**, instructions can be issued opportunistically to the execution units of processor **10** for execution without specific regard for data dependencies and anti-dependencies. The instructions, however, are maintained in issue queues **62-72** until execution of the instructions is complete and the result data, if any, are written back

to the rename registers associated with the GPRs **84**, **86** in case any of the instructions needs to be reissued.

As illustrated, the execution units of processor **10** include a CR unit (CRU) **90** for executing CR-modifying instructions, a branch execution unit (BEU) **92** for executing branch instructions, two fixed-point units (FXUs) **94** and **100** for executing fixed-point instructions, two load-store units (LSUs) **96** and **98** for executing load and store instructions, and two floating-point units (FPUs) **102** and **104** for executing floating-point instructions. Each of execution units **90-104** is preferably implemented as an execution pipeline having a number of pipeline stages.

During execution within one of execution units **90-104**, an instruction receives operands, if any, from one or more architected and/or rename registers within a register file coupled to the execution unit. When executing CR-modifying or CR-dependent instructions, CRU **90** and BEU **92** access the CR register file **80**, which in a preferred embodiment contains a CR and a number of CR rename registers that each comprise a number of distinct fields formed of one or more bits. Among these fields are LT, GT, and EQ fields that respectively indicate if a value (typically the result or operand of an instruction) is less than zero, greater than zero, or equal to zero. Link and count register (LCR) register file **82** contains a count register (CTR), a link register (LR) and rename registers of each, by which BEU **92** may also resolve conditional branches to obtain a path address.

General-purpose register files (GPRs) **84** and **86**, which are synchronized, duplicate register files, store fixed-point and integer values accessed and produced by FXUs **94** and **100** and LSUs **96** and **98**. Floating-point register file (FPR) **88**, which like GPRs **84** and **86** may also be implemented as duplicate sets of synchronized registers, contains floating-point values that result from the execution of floating-point instructions by FPUs **102** and **104** and floating-point load instructions by LSUs **96** and **98**. After an execution unit finishes execution of an instruction, the execution notifies GCT **38**, which schedules completion of instructions in program order.

The present invention is described with reference to the above data processing system **100** and processor **10A** but may be implemented in many other types of data processing system and processor architecture. The reference herein to a particular system architecture is therefore not meant to be limiting on the invention.

Referring now to **Figure 2**, there is illustrated a preferred embodiment of LSU **96**, **98** of **Figure 1** in accordance with the present invention. LSU **96**, **98** is one of the execution units within the processor core of processor **10A** illustrated in **Figure 1**. LSU **96**, **98** typically executes load instructions, which load data from L1 data cache **20**, L2 cache **16**, or memory **12** into selected general purpose registers (GPRs) **84**, **86**, GPR rename buffers, fixed purpose registers (FPRs) **88** or FPR rename buffers in the processor core. LSU **96**, **98** also executes store instructions, which store data from a

selected one of GPRs, GPR rename buffers, FPRs, or FPR rename buffers to memory. The present invention extends the functionality of the LSU during loading and storing of data to allow speculative loading and storing beyond a sync in the instruction sequence as well as speculative execution of other instruction types following the sync that may utilize the speculatively loaded (or stored) data. In the preferred embodiment, the speculatively loaded data and results from the subsequent speculatively executed instructions are stored within GPR and FPR rename registers of processor **10A** until the data is determined to exhibit correct dependencies..

Returning now to **Figure 2**, LSU **96, 98** includes adder **218**, which receives load/store instructions from an instruction sequencing unit (ISU) **200** via load/store request bus **217**. ISU **200** represents a collection of various components illustrated in **Figure 1**, which collectively provides instructions from the instruction cache **18**, L2 cache **16** or memory **12** to the other execution units of processor **10A** of **Figure 1**. The load/store instructions may be received in program order, i.e., in the sequence in which they were placed by the computer or programmer. Adder **218** calculates the target effective addresses of load and store instructions in the instruction stream. Adder **218** then forwards the target addresses for load instructions to load dispatch control unit **205** and forwards sync instructions and target addresses for store instructions to store/barrier queue **207** (illustrated as STQ **110** outside of LSU **96, 98** in **Figure 1**).

Load dispatch control unit **205** places the load instructions into an N entry (0 to N-1) Load Request Queue (LRQ) **208**. For simplicity an 8 entry (0-7) LRQ **208** is illustrated. Load register 0 **209** through load register 7 **211** hold the load instructions and are further coupled to L2 load request arbitration unit **213**, which determines the order for issuing the load requests out to L1 data cache **20** or L2 cache **16**. In the preferred embodiment, as illustrated, each load register has an affiliated multiprocessor speculation (MS) flag that indicates whether or not the load request is issued speculatively (i.e., subject to a previous sync operation). Thus, load register 0 **209** has an affiliated MS flag **210**, and load register 7 **211** has an affiliated MS flag **212**. Load requests that are subsequent to a sync in program order can be issued speculatively before the sync op completes on the system bus **14** when tracked by an affiliated MS Flag. In an alternate embodiment, where LRQ **208** issues load requests to L1 data cache **26** or L2 cache **30** sequentially, LRQ **208** may have a single MS flag and set the MS flag for the first load request issued prior to receipt of a sync ack.

In the preferred embodiment, LSU includes a barrier operation (BOP) controller **221**, which is coupled to both LRQ **208** and store/barrier queue **207**. BOP controller **221** is comprised of hardware logic by which the setting of the MS flag and other operations, which effectuate the speculative loads and subsequent execution of instructions according to the invention, are completed.

BOP controller **201** is coupled to L2 cache **16** via ack bus **214** by which sync acknowledgments are transmitted back to LSU **96**.

As described, BOP controller **221** maintains the MS flags, which mark speculative loads executed out-of-order with respect to previous syncs. BOP controller **221** monitors the store/barrier queue **207** to determine when a sync operation, which is sequentially ahead of issued load instruction(s) in the instruction sequence, has not been completed. MS flags **210**, **212** may be latches, which are set by BOP controller **221**. MS flags **210**, **212** may also be registers that include a single bit or group of bits depending on the desired functionality, as will become clear later.

MS flags **210**, **212** may be in "set" state when a load instruction is dependent on a previous sync, i.e., a speculative load, or in "reset" state when a load instruction is not dependent on a previous sync or the sync ack has been received at LSU **96**. When implemented as a single bit, MS flags **210**, **212** have a first set value (e.g. "1"), which represents to the LSU that the load is a speculative load, i.e., subject to receipt of a sync ack. MS flags **210**, **212** also have a second value (e.g. "0") that represents to LSU **96** that execution of the load does not depend on the receipt of a sync ack. BOP controller **221** and LRQ **208** control the issuance of the speculative loads and set the MS flags.

Transfer of instructions and data between the various components of **Figure 2** are completed with interconnect buses as shown. Thus, for example, both load dispatch control unit **205** and store/barrier queue **207** receive load and store/barrier instructions, respectively, from adder **218** via a core load/store request bus **217**, and data is returned from L2 cache **16** via load data return bus **215**. Further, system bus **14** provides a connection between L2 cache **16** and the external system components as provided in **Figure 1**.  
5  
10  
Although LSU **96** is depicted in **Figure 2** as having specific components, it is understood that additional components may be included within LSU including, for example, an address translation buffer. The presented figure is utilized for illustrative purposes only and is not meant to be limiting on the invention.  
15

The specific features and functionality of the LSU architecture are provided in the following commonly owned, cross-referenced, and co-pending patent applications: Serial No. 09/588,605 (Docket No. AT9-99-20 504) entitled "Multiprocessor Speculation Mechanism for Efficiently Managing Multiple Barrier Operations"; Serial No. 09/588,509 (Docket No. AT9-99-506) entitled "Mechanism for Folding Storage Barrier Operations in a Multiprocessor System"; Serial No. 09/588,483 (Docket No. AT9-99-507) entitled "Multi-level Multiprocessor Speculation Mechanism"; and Serial No. 09/588,507 (Docket No. AT9-99-508) entitled "System and Method for Providing Multiprocessor Speculation Within a Speculative Branch  
20  
25

Path". The relevant content of the above-referenced applications is incorporated herein by reference.

Unlike the architecture of the co-pending references, however, the present invention provides the data returned from the L2 cache **16** or other storage location directly to the processor's associated registers (or placed temporarily in L1 data cache **20**) for immediate use within the processor operations. In the preferred embodiment, the data from the load request is returned to the GPR or FPR rename register, i.e., the data is "written back," where it is made available for continuing processing by later instructions. Thus, the MS flag indicates the speculative nature of the load request in the LRQ **208** while the LSU **96** waits on the sync acknowledgment, but the data returned is made immediately available to the processor's execution units or registers before a sync ack is received by BOP controller **221**. Thus, the previous "wait on sync ack" requirement before executing subsequent load instructions (and following instructions) is removed, and load operations and other operations following the load are speculatively executed by the processor before the sync ack is received at the LSU.

When the sync ack is finally received, the BOP controller messages the affiliated logic of the rename registers that the data is no longer speculative and the affiliated logic resets the speculation flags associated with all instructions that followed the sync and/or utilized the data from the speculative load. If the data exhibits incorrect dependency or a snoop invalidate

returns, etc., the data and later values are discarded from the rename registers and the corresponding instructions are re-executed.

Figure 4 illustrates a sample rename register according to one embodiment of the invention. Rename register 400 is illustrated having 80 possible register entries 403. Rename register 400 includes a speculation flag 401, which identifies when the result stored within register entries 403 follows a yet-to-be completed barrier operation (i.e., are speculative). Rename register 400 also includes a GPR (FPR) number 405 associated with each register entry 403. GPR (FPR) number 405 indicates which entry of general purpose register 84, 86 (or FPR 88) the value stored in register entry 403 is assigned to when speculative flag 401 has a reset or off code. In a preferred embodiment, rename register 400 has an affiliated logic that receives messages from BOP controller 221 and sets/resets flag 401 for stored values accordingly.

Thus, when the data is initially returned and placed in the rename register 400, the affiliated logic sets the specific bit 401 to indicate that the data is speculative. Subsequent instructions that may have already been placed in the issue queues are monitored and the resulting values placed in the rename registers 400 are flagged (i.e., the bit 401 is set) by affiliated logic. In one embodiment, the affiliated logic may also set the bits within the issue queues of the instructions or the bits of the IISA instructions along with the bits

401 of the rename registers in which the resulting values are placed.

5 A sample instruction sequence with which the features of the present invention may advantageously be utilized is as follows:

A store addr(GPR30),data(GPR31)  
B load addr(GRR6),data(GPR18)  
C sync  
D load addr(GRR17),data(GPR12)  
E addition GPR18,x'0001'→GPR(20)  
F addition GPR12,x'0002'→GPR(14)  
G store addr(GPR5),data(GPR20)  
H store addr(GPR4),data(GPR14)  
I store addr(GPR1),data(GPR29)  
J XOR GPR18,GPR12→GPR27  
K branch to L if GPR27=X'0000', else go to Y  
L addition GPR18, x'0001'→GPR21  
M addition GPR12, x'0002'→GPR15  
N store addr(GPR3),data(GPR21)  
O store addr(GPR2),data(GPR15)  
P sync  
Q load addr(GPR8),data(GPR22)  
R addition GPR22, x'0003'→GPR23  
25 .  
Y (another set of instructions)  
30 .

Referring now to **Figure 5**, there is illustrated a timing diagram 500 for execution of the above sample instruction sequence (A - R) with full processor speculation as provided by the present invention. Eight time lines are provided, numbered 0-7, representing the total number of simultaneous operations possible within the processor. The actual time is provided as processor cycles and counted at line 0. Cycle 0 indicates start of processing for the above instruction sequence and cycle 281 indicates the time the last operation of the sequence completes.

At time 0, instruction A, B, D, and Q are executed. According to the instruction sequence, instruction D is launched prior to a preceding sync, and instruction Q is launch prior to two preceding syncs and a branch instruction. Instruction D thus exhibits a first level barrier speculation and has one associated barrier flag set. Instruction Q, however, exhibits a second level barrier speculation as well as a first level branch speculation. Thus instruction Q may have three corresponding speculation flags set, depending on the specific implementation of the invention utilized.

Data requested by instruction Q (Q data) returns at time 14, and instruction R and subsequent instructions, which utilizes Q data, commence execution at time 15. D data returns at time 17 cycles and instructions F and M, which utilizes D data, are executed at time 18 cycles. Thus, F is speculatively executed with respect to a barrier speculation, and M is speculatively executed with respect to both a barrier speculation and a branch

speculation and corresponding speculation flags are set for both instructions.

5 In operation, sync instructions are typically not executed until after a previous snoop on the bus completes. Thus, sync instruction C is executed at time 26 after a snoop response is received for instructions A and B at time 25. An acknowledgment is received for sync instruction C 100 cycles later at time 126. In the illustrated embodiments, sync operations requires about 100 processor cycles to complete in a 1 GHz SMP system.

10  
15 Once the sync ack returns, store instructions I and H are executed at time 127. At time 150 cycles, B data returns. Following, all instructions that depend on B data, i.e., instructions E, G, J and K, are executed in sequential order. Differences in the return time of data, e.g., B data and D data, may depend on which level of memory the data is located. Instruction D (load request) may have hit at L2 cache, while instruction B hit at lower level memory, resulting in a much longer response time.

20  
25 Instruction L is speculatively executed at time 153, prior to branch instruction K being resolved at time 154 cycle. Following, at time 155, instructions N and O are issued. Following the return of the snoop responses for instructions N and O at time 180, sync instruction P is issued. Associated sync ack then returns at time 281.

Thus, as illustrated with the above examples, with full processor speculation, all subsequent instructions

5 may be processed/executed immediately upon return of the load data before a preceding sync is even issued on the bus and/or before the sync ack returns. Present SMP commercial workloads exhibit correct "sync" speculation for greater than 99% of the operations across groups of instructions. Accordingly, if the speculative load is correct, i.e., data dependencies were observed, then a significant performance benefit may be achieved.

10 **Figure 3** illustrates the process by which speculative issuance of load instructions and subsequent operations are completed beyond issued barrier operations. As described above, the process involves barrier operation controller **221** setting MS flags, which may comprise multiple-bit registers or MS group flags.

15 The process begins at block **301** and thereafter proceeds to block **303**, where LSU **201** issues a sync operation on the system bus **14** and waits on a sync ack. Load dispatch control unit **205** places subsequent load requests in LRQ **208**. A determination is made by BOP controller **221**, at block **307**, whether any previously issued syncs operation have not completed on system bus **14**. When there are outstanding sync acks, the BOP controller **205** sets the flags of the load requests at block **309** to indicate that the loads are speculative. LRQ **208** then issues the load requests at block **309** to the cache and memory hierarchy.

20

25

A determination is made at block **311** whether or not data returned to LSU **201** from L1 data cache **20**, L2 cache **16**, or memory **12**. If data returned, then LSU **201** immediately forwards the data to processor registers or

5

10

15

20

25

30

execution units at block **311**, where the data may be utilized by instructions following the load in the instruction sequence. Once data arrives in processor registers, subsequent instructions that require the data are executed as illustrated in block **315**. Resulting values of the subsequent operations are generated and a determination is made at block **317** whether a sync ack has been received. If a sync ack has been received, then the values are placed into the GPR or FPR registers (i.e., the bit **401** in rename register **400** is not set as speculative) as illustrated in block **321**. If no sync ack has been received, however, the values are placed into rename registers **400** and affiliated logic sets the bits to indicate that the values are speculative at block **319**. Then the process ends at block **327**.

Returning to decision block **311**, the BOP controller **221** continues to monitor for the receipt of the sync ack at block **322**. If the sync ack has not been received, then a determination is made at block **323** whether the LSU **201** receives an invalidate. If an invalidate is received, then BOP controller **221** causes LRQ **205** to reissue the load request from the LRQ at block **325**. Also, at block **325**, BOP controller **221** messages the processor's execution units that any data previously provided by the load request is incorrect. The processor logic then handles correction of the operations that utilized the incorrect data. The correction may entail messaging the particular execution unit to which the data was provided to not utilize the data, removing the data and/or results from the respective register(s), and re-

executing all operations affected by the incorrect data at block 315. When the sync ack is received, all speculative flags (or bits) are reset as shown in block 323, values stored in the rename register become the values of corresponding GRPs and/or FPRs, and the processor continues execution of the instruction processes.

In the preferred embodiment of the invention, the speculative load functionality may be embedded within speculative branch prediction paths or vice-versa as indicated within the instruction sequence provided above. For example, the instruction sequence may contain branch within a speculative load instruction path, speculative loads within a branch, and multi-sync load speculation. With branch prediction, when the load data maintains correct dependency, substantial amounts of processing of the instruction sequence in the branch path is allowed to complete, resulting in faster processor operation. Thus, LRQ 208 continues to issue subsequent load requests, albeit speculatively, while waiting for both the sync acks and the determination of the correctness of the branch path. In the preferred implementation, both the load request and the speculative branch instructions have an associated bit(s) to identify to the processor's execution units that the instructions are speculatively executed and data are speculatively provided.

In one embodiment, the architecture of the IISA includes an appended bit or group of bits that are utilized to track the speculative nature of the instruction. The bit is provided a default value, for

example, "0" that indicates that the instruction is not speculative. A next value, e.g., "1" then indicates that the instruction is speculative. The setting of the value of the bit is controlled by speculative flag controller 160.

5

Also, a multiple bit embodiment is provided. Within an embedded speculatively executed branch path, a two bit flag may be set to 01 for a first load instruction and 11 for a second load instruction. The least significant digit "1" in each case represents that the particular instruction is a speculative load, while the most significant digit "0" and "1", respectively, indicates whether the load is being completed within a speculative branch. Thus, the first load instruction is not speculated within a speculative branch path, and the second load instruction is speculated within a speculative branch path. Alternatively, a speculative load within a speculative instruction sequence that is not a speculative branch path may utilize a similar multiple bit code.

10

15

20

25

In one preferred embodiment, the logic that controls instruction dispatching sets and controls a speculative bit of the IISA instruction to indicate that the instructions is speculatively issued following a yet-to-be completed barrier operation.

30

In a next embodiment, the bit or bits are localized within the queue associated with a particular speculatively executed instruction. Thus, each issue queue 62, 64, 66, 68, 70 and 72 may comprise setable bits

associated with each line of the queue and associated logic that sets and resets the bits to indicate whether the instruction is speculatively executed. The speculative bit may be set to a "1" or "0" to indicate whether or not the instruction is speculatively executed.

The present invention eliminates all throttling of the processor except for instances when throttling is absolutely necessary, such as to maintain correct data dependencies. Because the present processor architectures are designed with relatively deep queues, evaluating the correctness of the speculation occurs prior to the queue being filled and the data being committed to the processor's GPRs/FPRs, and the instructions may be re-issued from the queues without difficulty.

The present invention provides a new processor architecture (and method), which allows continuous issuing and execution of instructions, e.g., load instructions and subsequent instructions that may require the load data beyond a barrier operation in an instruction sequence. The processors utilized within the invention preferably provides greater than 99% accuracy with instructions executed beyond a sync as less than 1% of the data exhibit dependency on the prior execution of a previously issued instruction. The invention takes advantage of the high accuracy percentages within modern high frequency, mutiprocessor architectures to implement full-processor speculation, which removes previous holds on forward processing due to resolution of processor

speculation (if any) and results in increased overall processing speeds.

While illustrative embodiments have been particularly shown and described, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the illustrative embodiments.